Understanding Technical Debts of Requirements in an Industry Project: A Qualitative Study

Rhenara Oliveira[1,2][0009−0002−2501−9800], Anna B. Marques[1,2][0000−0001−9214−3399], Ismayle Santos[1,3][0000−0001−5580−643X], and Rossana Andrade[1,2][0000−0002−0186−2994]

[1] Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat), Ceará, Brasil
[2] Universidade Federal do Ceará, Ceará, Brasil
rhenaraalves21@alu.ufc.br, {beatriz.marques,rossana}@ufc.br
[3] Universidade Estadual do Ceará (Uece), Ceará, Brasil
ismayle.santos@uece.br

Abstract. Technical Debt (TD) in software engineering refers to the additional cost generated by inadequate decisions or implementations in software projects. Requirements TD emerges when specific requirements are overlooked, poorly understood, or implemented inadequately, leading to disparities between the developed product and the original specifications. In this context, requirements engineering should worry about how to prevent requirements TD from occurring and how to deal with requirements TD. This paper presents a qualitative study investigating the causes of requirements TD and identifying actions that can mitigate or resolve them. This qualitative study occurs in a software project of a partnership between an IT company and the Computer Networks, Software Engineering, and Systems Group (GREat) [4] of the Federal University of Ceará. The literature review helped identify types and causes of RTD, later correlated with responses from exploratory questions in project retrospective sessions. With the qualitative analysis conducted, significant challenges were detected, especially in the requirements elicitation phase, highlighting the need for more effective approaches. To deal with these issues, actions aimed at mitigating and resolving existing Requirements Technical Debts were identified and implemented.

Keywords: Requirements Engineering · Requirements Technical Debt · Qualitative Study.

1 Introduction

The development of software projects requires careful planning and execution to ensure the quality of the final product. However, various obstacles may arise, such as pressure for tight deadlines and limited resources, negatively impacting the software [8]. Teams sometimes rush through system development phases,

[4] https://www.great.ufc.br/

Proceedings of the 27th Workshop on Requirements Engineering (WER24), August 07-09, 2024, Buenos Aires, Argentina.
DOI: 10.29327/1407529.27-33

ignoring potential negative impacts, due to deadlines and budget constraints [10]. During the software development process, it is expected to identify issues that must be addressed to ensure the quality of the final product. However, neglecting to improve compromised tasks can result in Technical Debt (TD) caused by pending or inefficiently executed tasks [4].

Regarding Requirements Engineering, poorly conducted requirements elicitation can lead to errors that result in Requirements Technical Debt (RTD) [2]. These debts can arise intentionally when the requirements elicitation lead chooses not to follow the process correctly or unintentionally when team members need to gain more skill in the process [9].

In light of this scenario, the following research questions arose: How can RTD be prevented? How do we deal with these debts throughout the project? To address these issues and achieve the objective of this investigation, we conducted a qualitative study. The study explores the causes of RTD in software project collaboration between industry and academia, along with improving actions to mitigate them. This study involved retrospective sessions with the project's requirements team members. Data was categorized using qualitative analysis based on observed RTD types in literature.

In addition to this introduction, the article provides the main concepts related to Requirements Technical Debts. Next, the adopted methodological procedures are detailed, followed by the presentation and analysis of the results, from which the research conclusions are derived. As closure, final considerations are presented, providing suggestions for future investigations.

2 Background and Related Works

The emphasis on software quality assurance is one of the strengths of requirements engineering, as an inadequate understanding of requirements can lead to communication issues, rework, and dissatisfaction among end users. Therefore, dedicating time and effort to this stage is crucial for the success of any software development project [11].

Requirements Technical Debt (RTD) was first defined in 2012 and refers to the discrepancies between the final software product and its requirements specification [2]. These differences stem from inadequate decisions, resulting in low-quality elicitation and incomplete requirements, potentially leading to higher refactoring costs.

The presence of RTD in software development can cause delays and additional costs when requirements are not properly defined. Investing in the requirements elicitation and analysis phase is crucial to avoid these issues, and the adoption of agile practices can facilitate the efficient identification and resolution of requirements issues [1].

The work by Lenarduzzi and Fucci [5] defines RTD into three types:

- Type 0: Incomplete user needs. This occurs when the stakeholders needs are not fully met or understood, resulting in significant gaps in the documentation and understanding of the users functionalities and needs.

– Type 1: Requirements with "smells." Common when quality standards like ISO/IEC/IEEE 29148:2018 are violated during requirements elicitation. It can also be seen in other documentation styles. When unresolved, functionalities may be implemented with errors.
– Type 2: Incompatible implementation. This condition arises when the development team adjusts implemented functionalities to fix a requirement error, which results in an inconsistency with stakeholder expectations.

The study proposed by [7] investigates how the identification and measurement of Requirements Technical Debt are conducted in practice. Through a survey applied to 30 industry professionals, the study identifies causes of RTD, strategies, and metrics for its identification and measurement, as well as the difficulties faced by professionals. The results show a lack of specific knowledge and automated tools to assist in these activities, revealing the need for improvement in RTD management.

3 Methodology

This section presents the activities that make up the methodology (Figure 1). Initially, a literature review was carried out on the topic of the work to collect bibliographic data. Subsequently, retrospective sessions were held with members of the project requirements team to collect data. After the sessions, data was analyzed and categorized, resulting in the creation of a coding scheme based on the causes of RTD.

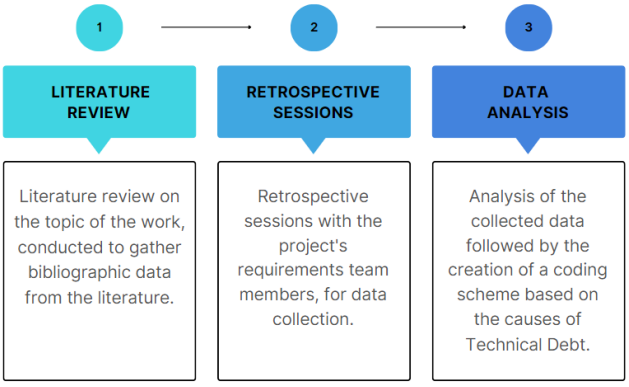

Fig. 1. Methodology

3.1 Literature Review

Initially, a literature review was conducted to understand the concept of RTD and its characteristics, causes, and types. For this purpose, searches were per-

formed in the *ACM*, *IEEE Digital Library*, *Scopus*, and *Science@Direct* databases, using keywords such as "Technical Debt Requirements" and "Requirements Engineering" from 2010 to 2022. Reading and analyzing the identified studies made it possible to determine the leading causes of RTD in software projects.

The literature review helped us create an initial framework for RTD types and causes. We used the identified causes as indicators for correlating data collected in later research phases.

3.2 Retrospective Sessions

This research was conducted in partnership between academia and industry, involving a leading computer hardware company. The goal was to develop a web platform to receive orders for customized products and suggest the best product or indicate the need for a new product to meet demand. The company will not be identified for confidentiality reasons.

The company adopts agile methodologies, and the project sprints lasted 15 days, with daily meetings and sprint reviews. The project team consisted of members from GREat-UFC and the company. All teams (requirements, UI/UX, development, and testing) were diverse in this regard. The project's product owners were two *Product Managers* from the company.

During the research period, the requirements team consisted of three analysts and two field researchers, who conducted the 1st retrospective session. However, there was turnover during this period, with two analysts and one researcher changing. The new researcher led the 2nd retrospective session.

The first retrospective session took place after the project's first release and included two researchers in the field and three requirements analysts involved in the project. Due to remote work, the session was conducted using Google Meet, Ideaboard for retrospective facilitation, and Miro for organizing the generated data. The meeting was recorded with the consent of the involved individuals, allowing for later review and manual transcription of the data. In the 1st retrospective, exploratory questions were adopted to analyze the potential existence of RTD:

- "What did we do well in the Requirements activities?"
- "What can we improve in the Requirements activities?"
- "What have we learned throughout the project?"
- "What will we do differently in the Requirements activities?"

Based on the collected data, a researcher categorized the responses into types and causes of RTD existing in the literature and validated them with the participants later.

The 2nd retrospective session was conducted remotely via Google Meet after the second project release. It involved a researcher specializing in RTD and two requirements analysts. This time, the same exploratory questions from the 1st retrospective were used. Still, the information was filled in a spreadsheet, along with the categorization of responses to the types and causes of TD found in the

literature. Subsequently, all data from both sessions were consolidated into a general spreadsheet.

Data generated during the sessions was analyzed and five main causes of RTD were identified. We referred to study conducted by [6] to identify causes and strategies for RTD identification and measurement, along with supporting metrics. Data was categorized based on responses and RTD causes/types.

3.3 Data Analysis

After collecting and organizing the data gathered in the two retrospective sessions into a spreadsheet, a detailed qualitative analysis was conducted.

Qualitative data analysis represents an approach to understanding and interpreting complex phenomena across various fields of study [3]. Unlike quantitative analysis, which focuses solely on numbers and measurements, qualitative analysis seeks a deeper exploration of meanings and contexts. This approach enables researchers to identify patterns that are not easily quantifiable, resulting in a more comprehensive understanding of the data [12].

To achieve the objectives of this research, the qualitative analysis focused on understanding and categorizing the observed causes of RTD in the project under study, in line with the general causes identified in the literature; that is, it adopted closed coding procedures [3]. The pre-established codes were types and causes of RTD identified in the literature review. Efforts were made to classify the activities carried out to mitigate or resolve the RTD throughout the project.

4 Results

Based on the qualitative analysis conducted, in which the causes identified were categorized, a data schema was possible to develop. This schema incorporates the general causes of RTD identified in the literature, the specific causes pointed out by the group of professionals involved in the project under study, and the improvement actions implemented to mitigate these debts.

Figure 2 presents the first schema, containing the data obtained in session 1. The following causes are identified: absent client, language divergence and inaccurate information, lack of defined process, inadequate requirement definition, significant changes in requirements, and incomplete requirements. To address these technical debts, the team adopted the following practices:

- **Increased involvement with stakeholders:** Increased interaction and collaboration with all project stakeholders through regular meetings.
- **Use of effective communication channels:** Use social media groups and other communication tools to ensure clear, accurate, and timely exchange of information among team members.
- **Team alignment:** Conduct daily meetings with the overall team to align priorities and track the progress of activities.
- **Adaptation to the Scrum method:** Implementation of Scrum practices and principles in project management and development.

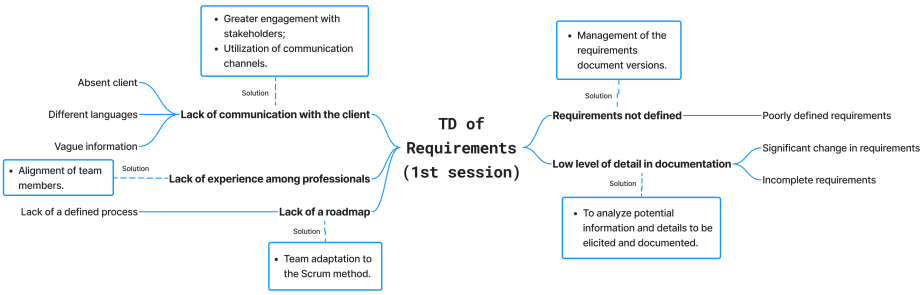

Fig. 2. Data Schema of Session 1

– **Version control of the requirements document:** Implementation of version control for project documentation, ensuring that all changes are recorded in an organized and accessible manner, facilitating traceability.
– **Thorough analysis of information and details to be documented:** Conduct a careful and detailed analysis of all relevant information and details to be included in the project documentation.

Figure 3 illustrates the second diagram conducted in session 2 at the end of the project. This diagram details the following causes: Non-completion of planned parts, stakeholder changes, project scope alteration, requirements team modification, low English proficiency, sprints with unclear objectives, outdated backlog, internal communication failure, lack of standardization in documentation writing, significant changes in requirements, and rework or delay in requirement elaboration. Furthermore, the actions implemented to mitigate the identified problems and debts were as follows:

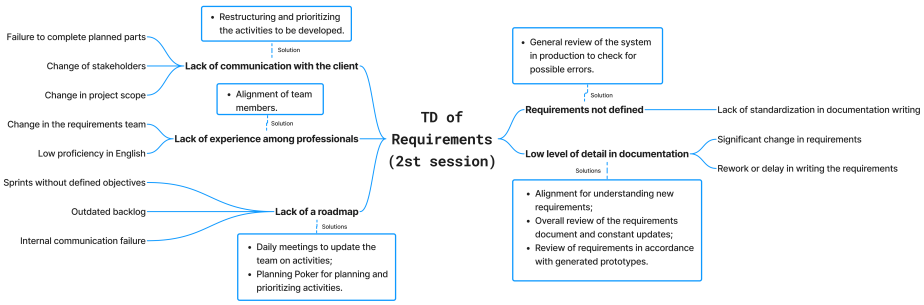

Fig. 3. Data Schema of Session 2

– **Restructuring and prioritizing activities to be performed:** Reorganizing and defining project activities, assigning priorities, aiming to optimize time and available resources.

- **Alignment of new team members:** Initial alignment meeting to ensure that new team members understand the project's objectives and processes.
- **Overall team adjustment:** Enhance team performance through conversations and dynamics, aiming to optimize collaboration and communication.
- **English courses:** Offering English courses for team members to improve communication skills and language comprehension.
- **Daily meetings:** Conduct short meetings with the team to discuss project progress, challenges, and priorities.
- **Use of planning poker for prioritization and planning of activities:** Utilization of the planning poker technique during activity planning, where the team estimates the effort required for each task, aiding in prioritizing tasks efficiently.
- **Complete review of the system in production, alignment with new requirements:** Thorough review of the production system to verify alignment with the new requirements.
- **Comprehensive review of the requirements document:** Detailed and thorough analysis of the project requirements document to ensure accuracy, clarity, and suitability.
- **Continuous documentation updates:** Continuous updating of the requirements documentation to accurately reflect the current project status.

The results demonstrate the importance of a detailed analysis of the causes and actions related to Requirements Technical Debt in software projects. Retrospective sessions revealed various challenges faced by the team, ranging from communication issues and changes in requirements to language domain and lack of defined processes.

According to feedback from participants in the retrospective sessions, actions to address these issues effectively mitigated technical debts and improved the software development process. Some strategies adopted were restructuring activities, team alignment, task prioritization, and investment in training, which increased efficiency and quality in the final product.

These findings highlight the importance of a proactive approach to managing RTD and the ongoing need for adapting and improving processes throughout the project life cycle. By recognizing and addressing challenges collaboratively, teams can ensure more robust software development aligned with customer expectations.

5 Conclusion

A comprehensive data schema was possible based on the qualitative analysis conducted, which categorized the causes identified by the requirements analysts. This schema incorporates the general causes of RTD described in the literature, the specific causes pointed out by the project team under study, and the actions implemented to address these issues.

Analyzing the causes and actions related to RTD is vital, as are continuous improvements in software development processes. Collaborative efforts to

address challenges can lead to a final product aligned with customer expectations. However, the study identifies areas for improvement, such as the need for more analysis through continuous monitoring using measures and indicators. Therefore, future research could explore this aspect, investigate the long-term effectiveness of implemented actions, and explore other methodologies for dealing with RTD.

Finally, I would like to thank the INCT INES 2.0 project (National Institute of Science and Technology for Software Engineering) for the support provided throughout the completion of this research.

References

1. Abad, Z.S.H., Ruhe, G.: Using real options to manage technical debt in requirements engineering. In: 2015 IEEE 23rd international requirements engineering conference (RE). pp. 230–235. IEEE (2015)
2. Ernst, N.A.: On the role of requirements in understanding and managing technical debt. In: 2012 Third International Workshop on Managing Technical Debt (MTD). pp. 61–64. IEEE (2012)
3. Gibbs, G.: Análise de dados qualitativos: coleção pesquisa qualitativa. Bookman Editora (2009)
4. Kruchten, P., Nord, R.L., Ozkaya, I.: Technical debt: From metaphor to theory and practice. Ieee software **29**(6), 18–21 (2012)
5. Lenarduzzi, V., Fucci, D.: Towards a holistic definition of requirements debt. In: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). pp. 1–5. IEEE (2019)
6. Melo, A., Fagundes, R., Lenarduzzi, V., Santos, W.: Identification and measurement of technical debt requirements in software development: a systematic literature review. arXiv preprint arXiv:2105.14232 (2021)
7. de Melo, A.C.C., Fagundes, R., Lima, J.V.V., Alencar, F., Santos, W.: Identificaçao e mensuraçao da dıvida técnica de requisitos: um survey na indústria de software. In: Anais do WER21-Workshop em Engenharia de Requisitos (2021)
8. Rios, N., de Mendonça Neto, M.G., Spínola, R.O.: A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. Information and Software Technology **102**, 117–145 (2018)
9. Rios, N., Spínola, R.O., Mendonça, M., Seaman, C.: Supporting analysis of technical debt causes and effects with cross-company probabilistic cause-effect diagrams. In: 2019 IEEE/ACM International Conference on Technical Debt (TechDebt). pp. 3–12. IEEE (2019)
10. da Silva Maldonado, E., Shihab, E., Tsantalis, N.: Using natural language processing to automatically detect self-admitted technical debt. IEEE Transactions on Software Engineering **43**(11), 1044–1062 (2017)
11. Sommerville, I.: Software Engineering. Pearson, Boston, MA, 10 edn. (2015)
12. Tuffour, I.: A critical overview of interpretative phenomenological analysis: A contemporary qualitative research approach. Journal of healthcare communications **2**(4), 52 (2017)